



---

# Algoritmos II

## Listas

Prof. Fernando Del Moro



## Listas

Assim como os vetores, tuplas e agora Listas, tem o objetivo de armazenar múltiplos valores em apenas uma (1) variável.

Apesar de não ser o mais comum, são indexados ou seja, seus elementos podem ser acessados através de índices.

Uma diferença essencial quanto a Tuplas é que listas não são "imutáveis".

## Listas

Diferente das tuplas:

Uma lista é uma estrutura totalmente adaptável e que permite uma série de operações e possui uma lista grande de operações embutidas que auxiliam os processos de manipulação.





---

## **Quando utilizar as Listas**

Por ser dotada de uma série de recursos e customizações uma lista pode ser utilizada sempre que houver um grande número de operações a serem realizadas sobre os dados.

Estas operações podem ser simples adições no final da lista mas também permitem inclusões em posições específicas, remoções, ordenações e pesquisas.



## Como criar Listas em Python

```
lista = ['uma', 'lista']
```

```
lista = [1,'lista',True,2.3]
```

```
lista = list() função built-in
```



---

## **Operações sobre Listas**

Diferentes das tuplas, que não podem ser modificadas, as operações possíveis em uma Lista são as mais diversas.

Além do processo clássico de percorrer, podemos inserir em qualquer posição, remover elementos, fatiar a lista e fazer vários processos de busca por elementos.



## Operações sobre Listas

O processo de percurso por índice, idêntico ao de Vetor e Tuplas

```
for cont in range(0, len(lista)):  
    print(lista[cont])
```



---

## Operações sobre Listas

Percurso com for...each

```
for item in lista:  
    print(item)
```





---

## Operações sobre Listas

Percurso com for...each e enumerate

```
for cont,item in enumerate(lista):  
    print(f'No indice {cont} existe o valor {item}')
```



## Operações sobre Listas

A adição de novos valores pode ser feita de várias formas:

- `range`
- `append(valor)`
- `insert(posição, valor)`

## Operações sobre Listas

A adição por 'range' cria uma lista de valores em uma sequência equivalente ao que ocorre no 'for':

```
lista = list(range(2,10))
```

irá resultar na seguinte lista

```
lista = [2,3,4,5,6,7,8,9]
```

Lembre que o 10 não entra no range

## Operações sobre Listas

Imagine a seguinte lista:

```
lista = ['maria', 'jose', 'pedro']
```

```
lista.append('joaquim') vai resultar em
```

```
lista = ['maria', 'jose', 'pedro', 'joaquim']
```

append adiciona no  
final da lista, sempre!

## Operações sobre Listas

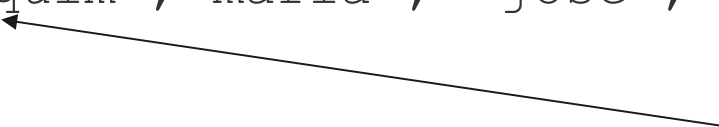
E o mesmo exemplo com *insert*:

```
lista = ['maria', 'jose', 'pedro']
```

```
lista.insert(0, 'joaquim')
```

 vai resultar em

```
lista = ['joaquim', 'maria', 'jose', 'pedro']
```



insert indica a  
posição onde será  
inserido

## Operações sobre Listas

Uma opção que agora pode ser utilizada é que quando um elemento não for mais necessário ele pode ser removido.



## Operações sobre Listas

Assim como na inserção, existem várias formas de se remover um elemento de uma lista:

<code>del lista[indice]</code>	→ Remove o elemento em <code>indice</code>
<code>lista.pop()</code>	→ Remove o último elemento
<code>lista.pop(indice)</code>	→ Remove o elemento em <code>indice</code>
<code>lista.remove(valor)</code>	→ Remove a primeira ocorrência de <code>elemento</code>

## Operações sobre Listas

Exemplos utilizando a lista abaixo:

[ 0                    1                    2                    3                    ]

```
lista = ['joaquim','maria', 'jose','pedro']
```

`del lista[2]` → Remove o elemento 'jose'  
`lista.pop()` → Remove o último, no caso 'pedro'  
`lista.pop(3)` → Remove o elemento 'pedro'  
`lista.remove('maria')` → Remove o elemento 'maria'



## Operações sobre Listas

Mas, pense....

[ 0 1 2 3 ]



```
lista = ['joaquim','maria', 'jose','pedro']
```

`del lista[5]` → Remove o elemento no índice 5 ?????

`lista.pop(32)` → pop no índice 32 ?????

`lista.remove('juca')` → Remove o elemento 'juca' ?????



---

## **Operações sobre Listas**

Naturalmente, qualquer um dos comandos anteriores vai causar um erro já que os elementos ou índices que estão sendo referenciados não existem.

Uma boa e sugerida ação é a verificação prévia se o elemento ou índice realmente existem.



## Operações sobre Listas

Se o índice é que será utilizado para a remoção (del ou pop(indice)) podemos utilizar a já conhecida instrução:

`len(lista)`



## Operações sobre Listas

Mas se a opção for pelo valor (`remove(valor)`) então existe a necessidade da utilização de um operador muito versátil:

**in**

Este operador retorna um *boolean* indicando a existência ou não de um valor dentro da estrutura.



---

## Operações sobre Listas

Exemplo:

```
if ('juca' in lista):  
    lista.remove('juca')
```



---

## Operações sobre Listas

Assim como em Tuplas é possível "fatiar" uma lista

```
lista[:]  
lista[indice:]  
lista[:indice]  
lista[indice:indice]
```

Lembrar que o que é retornado é uma outra lista, ou seja, existe uma "duplicação"



---

## Operações sobre Listas

Desta forma, se quisermos fazer uma cópia de alguma lista temos um procedimento a seguir:

```
lista = [1,2,3]
```

```
lista2 = lista #esta instrução não cria cópia
```

```
lista3 = lista[:] #esta instrução cria cópia
```



## Exercícios

1-Faça um programa que gere "N" números aleatórios entre -20 e 20, armazenados em uma lista e mostre:

- Quantos números negativos
- Quantos "0"
- Quantos Positivos

2-Crie um programa que leia vários números em uma lista (não perguntar quantidade).  
Mostre:

- Quantos valores foram digitados
- A lista de valores em ordem decrescente
- Se o valor "5" foi digitado





## Exercícios

3-Faça uma lista que pergunte ao usuário um número inteiro. Caso o número já exista, não adicione na lista. Ao final exibir todos em ordem crescente

4-Faça um programa que leia vários valores numéricos em uma lista. Mostrar o maior e o menor valor digitado e os índices onde eles estão. Considerar que o número pode ter sido digitado várias vezes



---

## Exercícios

5-Faca um programa que ajude um jogador da MEGASENA a criar palpites. O programa vai perguntar quantos jogos serão gerados e vai sortear 6 números entre 1 e 60 para cada jogo, cadastrando tudo em uma lista composta.

6-Crie um programa que leia Nome e duas notas de vários alunos guardando em uma lista composta. No final, mostre um boletim contendo a média de cada um e permita que o usuário possa mostrar as notas de cada aluno individualmente



---

## Exercícios

7-Crie um programa onde o usuário possa digitar "N" valores numéricos e cadastre-os em uma lista composta que mantenha separados os valores pares e ímpares. No final mostre os valores pares e ímpares em ordem crescente